A Performance Analysis of Symmetric Block Ciphers: AES, Camellia, and SM4

Edoardo Simonetti Spallotta
Student ID: 2077561
Sapienza University of Rome / ECSAI
`simonettispallotta.2077561@studenti.uniroma1.it`

Abstract

This paper presents a comprehensive benchmark study examining the performance characteristics of three major symmetric block ciphers: AES-128-CBC, Camellia-128-CBC, and SM4-CBC.[1] I developed a custom benchmarking tool in C++ using the OpenSSL library to obtain an empirical assessment of how these algorithms perform under various workload conditions. The primary objective was to measure their computational efficiency with precision, which I achieved through a carefully designed methodology that employed `std::chrono::steady_clock` for monotonic timing to avoid system clock irregularities, while restricting measurements exclusively to the core cryptographic operations. The experimental protocol included warm-up iterations to stabilize CPU cache behavior, followed by multiple timed runs from which I calculated mean execution times and standard deviations to ensure statistical reliability. Performance was evaluated across small, medium, and large data files to understand how these ciphers handle different operational scenarios, with particular emphasis on mean execution time, standard deviation, and throughput measured in MB/s for bulk encryption tasks. This report documents the complete methodology, presents the experimental findings, and discusses their implications for practical cipher selection.

1 Introduction

Symmetric encryption forms the backbone of modern data security, protecting information at rest on storage devices, in transit over networks, and within messaging systems. The fundamental principle is straightforward: a single shared key is used to both encrypt and decrypt data, making these algorithms computationally efficient compared to their asymmetric counterparts. However, the landscape of symmetric ciphers is diverse, with algorithms emerging from different standardization bodies and designed with varying priorities, which naturally leads to distinct performance characteristics that can significantly impact system design decisions.

When architecting security infrastructure, the selection of an appropriate cipher becomes a critical consideration, as different scenarios demand different trade-offs. In some cases, such as high-throughput data center operations or real-time video encryption, maximizing bulk data processing speed is paramount; in other contexts involving frequent encryption of small messages or packets, the initialization overhead of the cipher might dominate the total computational cost, making algorithm efficiency at small scales more relevant. These performance variations played a crucial role in motivating an empirical comparison of three prominent ciphers deployed in contemporary systems.

For this study, we selected three algorithms that represent different design philosophies and regional standards:

- **AES:** The globally dominant standard adopted by NIST in 2001, which has become ubiquitous in commercial and government applications worldwide.
- **Camellia:** A Japanese cipher developed collaboratively by NTT and Mitsubishi, widely regarded as a technically strong alternative to AES with comparable security properties.
- **SM4:** The Chinese national cryptographic standard, which has gained increasing prominence in hardware implementations and is mandatory for certain applications within Chinese regulatory frameworks.

The objective of this project was to establish a rigorous and equitable benchmarking framework that would yield reliable performance data across these three algorithms. I implemented the benchmark in C++ to maintain low-level control over the measurement process, leveraging the OpenSSL library for cryptographic operations to ensure I was testing well-vetted, production-grade implementations. This paper documents the complete experimental methodology, details the technical implementation, presents the quantitative findings, and offers analysis on the practical implications for cipher selection in real-world deployments.

2 Background on Ciphers

Before examining the experimental results, it is useful to establish the technical context for the three algorithms under evaluation. All three ciphers share fundamental design characteristics: they are 128-bit block ciphers that process data

[1]The complete source code, build instructions, and raw data are available at: https://github.com/prollyyes/OpenSSL-Symmetric-Cipher-Benchmark-Tool

in 16-byte blocks, and for consistency in the experiments, I configured each cipher to operate with a 128-bit key in CBC (Cipher Block Chaining) mode, which provides semantic security through the use of an initialization vector.

2.1 AES (Advanced Encryption Standard)

The Advanced Encryption Standard occupies a central position in contemporary cryptography and warrants particular attention. It was established as the U.S. federal standard in 2001 following an open competition conducted by NIST (National Institute of Standards and Technology), in which the Rijndael algorithm, designed by Belgian cryptographers Joan Daemen and Vincent Rijmen, emerged as the winner from among fifteen candidates.

AES employs a Substitution-Permutation Network architecture, which has proven to be highly efficient for implementation in both software and dedicated hardware. One of the most significant factors in AES's dominant market position is the widespread availability of hardware acceleration in modern processors since most contemporary CPUs include specialized instruction sets (such as AES-NI on x86 processors) that dramatically accelerate AES operations, making it exceptionally fast compared to ciphers that lack such hardware support. The standard supports key sizes of 128, 192, or 256 bits, and has effectively become the default choice for symmetric encryption across virtually all application domains.

2.2 Camellia

Camellia emerged from a collaborative development effort between NTT and Mitsubishi Electric in Japan around the year 2000, with the explicit goal of creating a cipher that could match or exceed AES in both security strength and implementation efficiency. Unlike AES, Camellia is built on a classic Feistel network structure, representing a fundamentally different design philosophy that offers certain theoretical advantages in terms of security analysis and the reversibility of encryption operations.

The cipher shares AES's block size of 128 bits and similarly supports variable key lengths of 128, 192, and 256 bits, providing flexibility for different security requirements. Camellia has undergone extensive cryptanalytic scrutiny and has been endorsed by multiple international standards organizations, including adoption as an ISO/IEC standard and recommendation by the European NESSIE project. Within the cryptographic community, it is often viewed as a robust alternative to AES, serving as a form of algorithmic diversity that would provide systems with a fallback option should any critical vulnerability ever be discovered in AES itself.

2.3 SM4

SM4 represents a relatively recent addition to the international cryptographic landscape, having been officially published by the Chinese government as a commercial cryptographic standard in 2012 as part of a broader initiative to establish indigenous cryptographic algorithms independent of Western standards. Prior to its public release, the algorithm was known as SMS4 and had been used in Chinese wireless LAN security protocols.

The cipher operates on 128-bit blocks with a fixed 128-bit key size, employing an unbalanced Feistel network structure that iterates through 32 rounds, substantially more than the 10 rounds used by AES with equivalent key length. This design choice reflects a conservative approach that prioritizes security margin, potentially at the expense of computational efficiency in software implementations. While SM4 has not achieved the global ubiquity of AES, it is increasingly being integrated into hardware platforms and cryptographic libraries, and its use is mandated for certain applications within Chinese regulatory frameworks, particularly in sectors involving sensitive government or financial data.

3 Benchmarking Methodology

Establishing a reliable performance benchmark requires meticulous attention to measurement methodology, as poorly designed experiments can produce misleading results that are worse than having no data at all. The approach chosen was designed to minimize sources of measurement error while ensuring that the results would be representative of real-world performance characteristics.

3.1 Test Data

Recognizing that cipher performance often varies significantly with input size due to factors such as initialization overhead, cache behavior, and the amortization of setup costs over larger datasets, I evaluated each algorithm across three distinct file sizes that represent different operational scenarios:

- **A 16-byte file:** This minimal size, corresponding to exactly one cipher block, allows us to assess the performance characteristics when initialization overhead dominates and to understand the baseline cost of a single encryption operation.
- **A 20 KB text file:** This intermediate size represents typical small payloads such as configuration files, API responses, or individual network packets, providing insight into performance at scales commonly encountered in application-level encryption.
- **A 2.5 MB binary file:** This larger file enables measurement of sustained throughput for bulk encryption tasks such as file system encryption or streaming data protection, where initialization costs become negligible relative to the total processing time.

To ensure experimental reproducibility, our benchmarking tool automatically generates these test files with precise byte counts and stores them in a `data/` directory, eliminating any dependency on external file preparation.

3.2 Measurement Process

The precision and validity of the timing measurements were achieved through careful selection of timing mechanisms and strict control over what operations were included in the measured intervals. I employed std :: chrono :: steady_clock from the C++ standard library, which provides a monotonic clock that advances at a consistent rate independent of system clock adjustments, making it specifically suitable for measuring elapsed time intervals without concern for clock synchronization events or daylight saving time changes.

Equally important was my decision to measure only the pure cryptographic operations, deliberately excluding auxiliary operations such as file I/O, memory allocation, and data structure management that would otherwise contaminate the timing data with system-dependent overheads unrelated to cipher performance. To achieve this isolation, I positioned timing calls immediately before EVP_EncryptInit_ex and immediately after EVP_EncryptFinal_ex, ensuring that only the OpenSSL encryption routines themselves contributed to the measured duration.

To further enhance the stability and reliability of my measurements, I implemented two additional methodological refinements:

1. **Warm-up Run:** Prior to collecting any timed measurements, I executed a complete encryption and decryption cycle for each cipher-file combination. This warm-up phase serves to populate CPU caches with relevant code and data, stabilize CPU frequency scaling mechanisms, and allow the operating system's thread scheduler to reach equilibrium, thereby preventing the first measured iteration from exhibiting artificially inflated latency due to cold-start effects.

2. **Multiple Iterations:** Rather than relying on a single measurement, I executed each benchmark scenario five times in succession, computing both the arithmetic mean and the standard deviation from this sample. The mean provides my primary performance metric, while the standard deviation quantifies measurement stability. Low standard deviation values indicate that my results are reproducible and not dominated by random system noise or scheduling variations.

3.3 Performance Metrics

The experimental protocol captured three primary performance indicators for each cipher-file combination, which together provide a comprehensive characterization of operational efficiency:

- **Mean Execution Time (ms):** The arithmetic mean of the five timed iterations, representing the expected latency for encrypting the given data size under typical conditions.

- **Standard Deviation (ms):** A measure of timing variability across the five runs, which serves as an indicator of measurement stability and allows assessment of whether performance is consistent or subject to significant run-to-run fluctuation.

- **Throughput (MB/s):** Particularly relevant for the large file test, this metric expresses processing rate in megabytes per second and is computed by dividing the file size by the mean execution time, providing an intuitive measure of bulk encryption capacity.

3.4 Correctness Verification

Performance measurements are meaningless if the implementation produces incorrect results, so I incorporated a verification step to ensure cryptographic correctness throughout all experiments. Following each series of timed encryption runs, the tool performed a complete decryption of the generated ciphertext and conducted a byte-by-byte comparison with the original plaintext input. Any discrepancy would trigger an immediate termination with an error report, ensuring that only collected timing data from correctly functioning implementations was retained. Across all experiments conducted for this study, no correctness failures were encountered, confirming the integrity of both the OpenSSL library implementations and our measurement harness.

4 Implementation

The benchmarking tool was implemented in C++17, leveraging modern language features for improved code clarity and safety, with CMake serving as the build system to ensure portability across different platforms. The only external dependency is the OpenSSL cryptographic library, which provides well-tested, production-grade implementations of all three ciphers under evaluation.

The codebase was structured using a clean separation of concerns, dividing functionality into two primary modules:

- bench.cpp: This module serves as the main driver program, orchestrating the complete experimental workflow including test file generation, execution of the benchmarking loops, statistical analysis of timing data, and serialization of results to a CSV file for subsequent analysis and visualization.

- crypto_utils .cpp: This abstraction layer encapsulates all interactions with the OpenSSL EVP interface, exposing clean, purpose-built functions such as encrypt_with_timing and decrypt_with_timing that handle the complexities of the EVP API while embedding the high-resolution timing logic. This design keeps the main benchmarking code focused on experimental logic rather than cryptographic implementation details.

To ensure consistency across all measurements and eliminate key generation as a confounding variable, a single encryption key and initialization vector are generated at program startup using OpenSSL's cryptographically secure RAND_bytes function, and these same values are reused throughout all subsequent test iterations, allowing direct performance comparison without variation in cryptographic parameters.

5 Results and Discussion

The complete benchmark suite was executed on a MacBook Pro equipped with an Apple M2 Pro processor, representing a contemporary ARM-based system architecture with modern cryptographic capabilities. Raw timing data from all experiments was systematically recorded to `results/benchmark_results.csv`, and subsequently processed using a Python visualization script that generated the graphical presentations shown in Figures 1 and 2.

5.1 Throughput Analysis

Examination of Figure 1, which presents throughput measurements for the 2.5MB file test, reveals a clear performance hierarchy among the three ciphers. AES demonstrates overwhelmingly superior throughput, processing bulk data at rates that significantly exceed those of the other two algorithms: a result that aligns with expectations given the widespread availability of dedicated hardware acceleration for AES in contemporary processor architectures. The M2 Pro's ARM cores include cryptographic extensions specifically optimized for AES operations, allowing the algorithm to leverage specialized silicon that can execute multiple encryption rounds per cycle, effectively making AES orders of magnitude faster than what would be achievable through pure software implementation.

Camellia occupies an intermediate position in the performance spectrum, achieving throughput that, while substantially lower than AES, remains quite respectable in absolute terms and would be more than adequate for most practical applications. The cipher's performance demonstrates that it represents a viable alternative when hardware-accelerated AES is unavailable or when algorithmic diversity is desired for security reasons, though the performance gap does highlight the significant advantage that dedicated instruction set extensions provide.

SM4 exhibits the lowest throughput of the three algorithms under test, which can be attributed to multiple factors inherent in its design. The cipher's use of 32 rounds (more than three times the number employed by AES with comparable key length) naturally increases the computational cost per block, and this overhead is compounded by the absence of widespread hardware acceleration support in current mainstream processors. In a pure software implementation environment, which is what our benchmark effectively measures on this platform, SM4's conservative security margin comes at a measurable performance cost relative to its competitors.

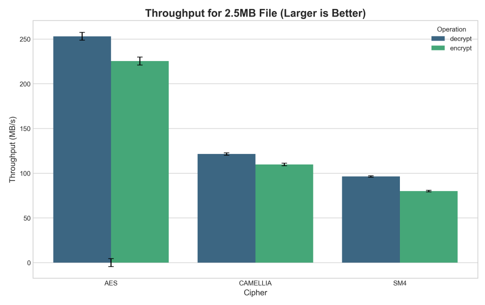

Figure 1: Throughput comparison for the 2.5MB file.

5.2 Performance Across File Sizes

Figure 2 presents execution times across all three file sizes on a logarithmic scale, revealing nuanced performance characteristics that vary with workload scale. The logarithmic presentation is particularly important here, as it allows simultaneous visualization of measurements spanning several orders of magnitude while making proportional differences comparable across different scale ranges.

The most striking observation emerges from the 16-byte file results, where the performance differential among the three ciphers is substantially compressed compared to the bulk encryption scenario. This convergence occurs because, at such small scales, the cipher initialization overhead – including context setup, key schedule computation, and the EVP framework's internal bookkeeping– represents a significant fraction of the total execution time. When encrypting only a single block, the actual encryption operation is so brief that these fixed costs dominate the measurement, effectively masking the per-block performance differences that become apparent at larger scales. This finding has practical implications for applications that frequently encrypt very small messages, suggesting that the choice of cipher may be less performance-critical in such scenarios than in bulk encryption contexts.

As we transition to the 20KB file size, which represents a more typical application payload, the performance hierarchy reasserts itself more clearly, with the ranking mirroring what we observed in the throughput analysis: AES maintains its lead, Camellia follows at a moderate distance, and SM4 trails in third position. At this intermediate scale, initialization costs have been largely amortized across the larger number of blocks, allowing the per-block computational efficiency of each algorithm to become the dominant factor in overall execution time.

An important indicator of measurement quality visible in both figures is the standard deviation, represented by error bars on each data point. These error margins remain notably small across all experimental conditions, typically representing only a small fraction of the mean values, which provides strong evidence that our measurements are stable, reproducible, and not significantly contaminated by system noise or timing variability. This consistency validates our methodological choices regarding warm-up iterations and multiple measurement samples.

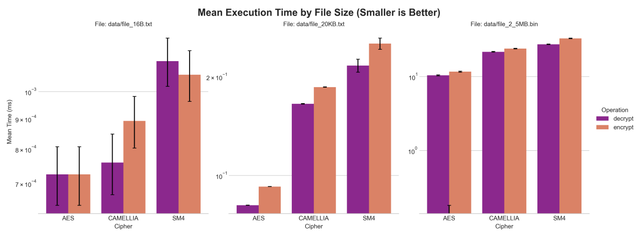

Figure 2: Performance comparison across all file sizes (log scale).

6 Conclusion

This project successfully established a rigorous benchmarking framework for comparing the performance characteristics of three prominent symmetric block ciphers, yielding reliable empirical data that can inform practical cipher selection decisions. Through careful attention to measurement methodology, including the use of monotonic timing sources, isolation of pure cryptographic operations, warm-up iterations, and statistical sampling, we obtained performance metrics that are both precise and reproducible, as evidenced by the consistently low standard deviations observed across all experimental conditions.

The experimental findings strongly confirm the prevailing understanding within the cryptographic engineering community: AES maintains a commanding performance advantage on modern hardware platforms, with throughput that substantially exceeds that of competing algorithms when hardware acceleration features are available. This dominance is not merely incremental but represents a qualitative difference that makes AES the natural first choice for performance-critical applications. Camellia, while unable to match AES's accelerated performance, nonetheless demonstrates solid computational efficiency that would be entirely adequate for most practical use cases, and its status as a well-analyzed alternative to AES gives it value in scenarios requiring algorithmic diversity. SM4, despite its lower relative performance in software-only implementations, serves important roles within its target deployment contexts and may see performance improvements as hardware support becomes more widespread in processor designs targeting Asian markets.

It is important to recognize that cipher selection in real systems involves considerations that extend beyond raw computational performance. Security requirements, regulatory compliance, hardware platform capabilities, and regional standardization mandates all factor into the decision-making process, and different operational contexts may weight these factors differently. The performance data presented in this work provides one essential input to that multifaceted decision process, offering quantitative evidence that can be weighed alongside other technical and policy considerations when architecting secure systems. The hope is that this dataset contributes to more informed engineering choices in the deployment of cryptographic infrastructure.

References

1. National Institute of Standards and Technology (NIST). (2001). *Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication 197.
   Available at: https://doi.org/10.6028/NIST.FIPS.197

2. NTT and Mitsubishi Electric Corporation. *Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms.*
   Available at: https://info.isl.ntt.co.jp/crypt/eng/camellia/

3. Specification of SMS4, Block Cipher for WLAN Products — SM4. (2006). *SM4 Block Cipher Algorithm.* Office of State Commercial Cryptography Administration, China.
   Available at: https://en.wikipedia.org/wiki/SM4_(cipher)

4. OpenSSL Software Foundation. *OpenSSL Cryptography and SSL/TLS Toolkit.*
   Available at: https://www.openssl.org/

5. Simonetti Spallotta, E. (2025). *OpenSSL Symmetric Cipher Benchmark Tool.* GitHub Repository.
   Available at: github.com/prollyyes/OpenSSL-Symmetric-Cipher-Benchmark-Tool.git

Appendix: Complete Benchmark Results

The following table presents the complete raw data collected during the experimental campaign, including all timing measurements and throughput calculations for both encryption and decryption operations across all cipher-file combinations tested.

Cipher	Operation	File	Size (B)	Mean (ms)	StdDev (ms)	Throughput (MB/s)
AES	Encrypt	16B	16	0.000725	0.000063	22.08
AES	Decrypt	16B	16	0.000725	0.000082	22.07
AES	Encrypt	20KB	20,480	0.092442	0.001326	221.54
AES	Decrypt	20KB	20,480	0.081125	0.000087	252.45
AES	Encrypt	2.5MB	2,621,440	11.629200	0.135601	225.42
AES	Decrypt	2.5MB	2,621,440	10.358475	0.180987	253.07
Camellia	Encrypt	16B	16	0.000892	0.000136	17.95
Camellia	Decrypt	16B	16	0.000759	0.000089	21.09
Camellia	Encrypt	20KB	20,480	0.185833	0.011916	110.21
Camellia	Decrypt	20KB	20,480	0.165217	0.000318	123.96
Camellia	Encrypt	2.5MB	2,621,440	23.888150	0.125349	109.74
Camellia	Decrypt	2.5MB	2,621,440	21.588000	0.248705	121.43
SM4	Encrypt	16B	16	0.001067	0.000224	15.00
SM4	Decrypt	16B	16	0.001125	0.000105	14.22
SM4	Encrypt	20KB	20,480	0.252125	0.005145	81.23
SM4	Decrypt	20KB	20,480	0.216192	0.009845	94.73
SM4	Encrypt	2.5MB	2,621,440	32.786133	0.115911	79.96
SM4	Decrypt	2.5MB	2,621,440	27.218475	0.245050	96.31

Table 1: Complete benchmark results showing mean execution time, standard deviation, and throughput for all cipher-operation-file combinations. All measurements represent the mean of 5 runs following a warm-up iteration.

Notes:

- All measurements were conducted on a MacBook Pro with Apple M2 Pro processor.
- Each data point represents the arithmetic mean of 5 independent runs.
- Standard deviation quantifies measurement stability across runs.
- Throughput is calculated as file size divided by mean execution time.
- All ciphers operated in CBC mode with 128-bit keys.